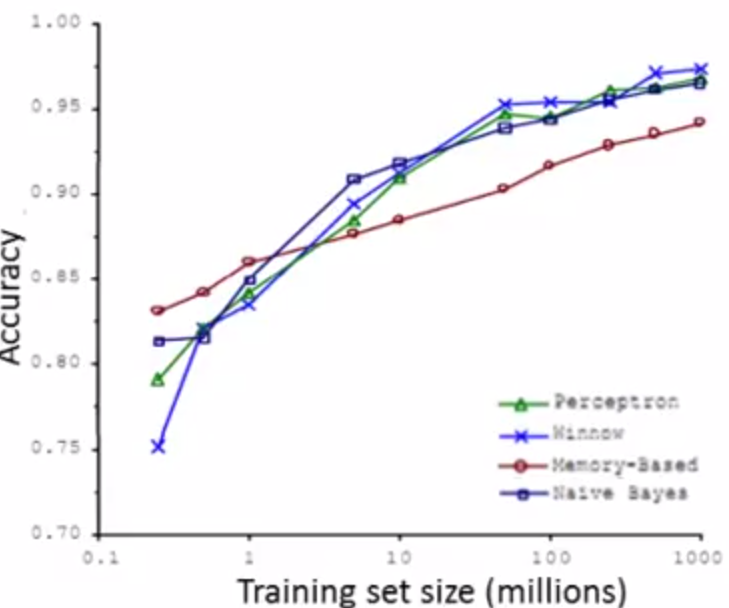
***Using Large Datasets***

**I. DATA FOR MACHINE LEARNING**

* Under certain conditions, getting a lot of data + training on a certain type of learning algorithm can be a very effective way to get a learning algorithm have very good performance.
* This arises often enough that if those conditions hold true for your problem + if you're able to get a lot of data, this could be a very good way to get a very high performance learning algorithm.
* Years ago, 2 researchers, Michelle Banko + Eric Broule, ran the following fascinating study.
* They were interested in studying the effect of using different learning algorithms vs. trying them out on different training set,
* They were considering the problem of classifying between confusable words
* “For breakfast I ate” 🡪 to, two or too?



* They took supervised ML problems like these + tried to categorize what is the appropriate word to go into a certain position in an English sentence.
* They used a few different learning algorithms considered state of the art at the time, a variance on logistic regression called **Perceptron,** a **Winnow** (similar to logistic regression in some ways), a **memory-based learning algorithm**, + a **Naïve Bayes algorithm**,
* So w/ 4 different classification algorithms, they varied the training set size + tried out these algorithms on a range of training set sizes

T

* The results trends are very clear: These algorithms give remarkably similar performance.
* Also, as the training set size increases (in millions), performance of the algorithms all pretty much monotonically increase
* In fact, if you pick any algorithm that be an "**inferior algorithm**" but if you it more data, then from these examples, it looks like it will most likely beat even a "**superior algorithm**".
* This original study was very influential, + since it was conducted, there've been a range of many different studies showing similar results
* Many different learning algorithms can sometimes, depending on details, can give pretty similar ranges of performance, but what can *really drive* performance is giving an algorithm a ton of training data
* Results like these led to a saying in ML: “It's not who has the best algorithm that wins, it's who has the most data”
* When is this true + when is this not true?
* B/c w/ a learning algorithm for which this is true, then getting a lot of data is often the best way to ensure we have an algorithm w/ very high performance
* Let's try to lay out a set of assumptions under which having a massive training will be able to help.
* Assume the features x1-x(n) have sufficient info w/ which we can use to predict y accurately.
* For example, if we take the confusable words + say that its features X capture the surrounding words around the blank that we're trying to fill in.
* Here, that is pretty much the info information that tells us the word we want in the middle is TWO, not TO nor TOO.
* These features give us enough info to unambiguously decide what the label y is
* Counter example: Predicting price of a house only from size
* There's so many other factors that affect price of a house other than just size
* If all you know is size, it's actually very difficult to predict price accurately.
* Can test 🡪 Ask yourself: Given the input features X, if we were to go to human expert in this domain, can a they confidently predict the value of y.
* For this first example if we go to, you know an expert human English speaker, they’d be able to predict what word should go in here,
* This gives me confidence that X allows us to predict Y accurately
* In contrast if we go to a human expert in housing prices, like an expert realtor, + just tell them the size of a house + they wouldn't be able to tell me the price
* For the housing price example, knowing only size doesn't give enough info to predict price
* Large Data Rationale
* Suppose the assumption that the features X have enough info to predict the value of Y holds
* Now suppose we use a learning algorithm w/ a large number of parameters/features, or a NN w/ many hidden units
* These powerful learning algorithms w/ a lot of parameters can fit very complex functions
* Think of these as low-bias algorithms b/c we can fit very complex functions
* Chances are, if we run these algorithms on data sets, they’ll will be able to fit training well, + so hopefully training error will be small.
* If we use a massive training set, then hopefully, even w/ a lot of parameters, the training set is even larger than the number of parameters + then these algorithms will be unlikely to overfit the training data
* The training error will hopefully be close to test error.
* Finally putting these 2 together (training error is small + test error is close to training error), implies that the test error
* Another way to think about this is that in order to have a high-performance learning algorithm, we want it not to have high bias nor high variance.
* So, the bias problem we address w/ a learning algorithm w/ many parameters
* And using a very large training set ensures we don't have a variance problem
* This allows us to do well on the test set.
* Fundamentally, it's key ingredients:
* **Assuming the features have enough info**
* **We have a rich class of functions that guarantees low bias**
* **We have a massive training set to guarantee low variance.**
* A problem w/ a lot of data + training a learning algorithm w/ lot of parameters might be a good way to give a high-performance learning algorithm
* The key test is “can a human expert look at the features X + confidently predict the value of Y”.
* This is a certification that Y can be predicted accurately from the features X
* Also, ask if you *actually* get a large training set, + train the learning algorithm w/ *a lot* of parameters in the training set
* If you can do both then that's more often give you a very kind performance learning algorithm.